BITS F464: Machine Learning

09 Curse of Dimensionality PCA and Eigenfaces

Dr. Kamlesh Tiwari
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

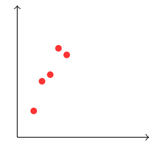
Feb 05, 2021 ONLINE (Campus @ BITS-Pilani Jan-May 2021)

http://ktiwari.in/ml

Curse of Dimensionality

Observed vs True Dimensionality

- Consider following data and answer what is its dimensionality?
- Is it not 2?



- May be not!
- Data may be observed across different sensors, so small variation maybe due to noise or ...
- It could also be possible that all the observations may be dependent on some quantity which is not being measured

Databases are generally of high dimension: Images contain lot of pixels and text have lot of words (250×250 pixels, or 10^6 words)

Curse of Dimensionality

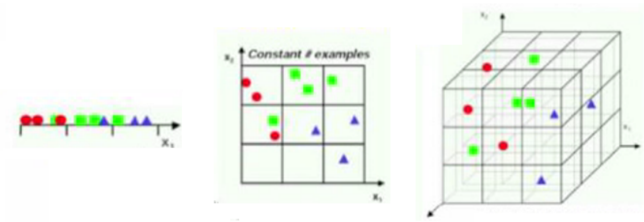
Consider handwritten digits

- Assume 20×20 bitmap (2^{400} observation)
- We would never see most of the events
- True dimensionality is something like number of possible pen strokes

Curse of Dimensionality

Why it is a problem

- Because most of the machine learning methods tends to **count** evidences
- Space grows very quickly but number example remains limited
- Density of data decreases sharply with the dimension. This lead to no observations for most of the cases



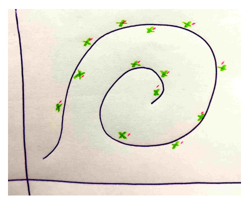
Data Lies on Edge

- Consider n data points in d dimension. Scale values to [0-1]
- Find side σ of minimal hyper-cube having k nearest neighbours
  Assume data is uniformly distributed in the unit space. Volume σ^d would have k/n share of all points. So $\sigma = (k/n)^{1/d}$. With $k = 10$, $n = 1000$ the value of σ for $d = 2, 10, 100$ are $0.1, 0.63, 0.95$. Nearest points are far away for large d. **This kills the notion of k near points** as near and far points are at the same place.
- Another way: consider a point to be in interior of the hyper cube, if it is ϵ distance apart from both the ends 0 and 1.
  - What is the probability a point would be in interior? $(1 - 2\epsilon)$
  - There are d dimensions. What is the probability a point would be in interior for all dimensions? $(1 - 2\epsilon)^d$
  - $(1 - 2\epsilon)$ is less than 1 (let 0.9). For $d = 100$, $(1 - 2\epsilon)^d = 0.000026$. Only 26 points out of 1000000 lies in 0.9 interior.

In high dimensional space, most of the data points lies on edge.

Manifold for Curse of Dimensionality

- True dimensionality may be lot lower than the observed one.
- Data may be in some low dimensional manifold (sheet) in high dimensional space



- There is a possibility that in the high dimensional space, there lies a sub-space (manifold) where most of the data resides.
- The manifold could be curved so as to explore all the dimensions
- Assumption is that the data point never leaves the manifold
- Distance is computed on manifold surface, that is low dimension

We are on Earth's sphere but, locally is is flat.

Count number of point in σ and 2σ neighbourhood. If points increases 4 times; it is a 2D manifold. If it is 9 times it is a 3D. And so on.

Mean, Variance and Covariance

- Conceptually **mean** represents the center and **variance** the spread of data points
- Let $a = [a_1 a_2 .. a_n]$ and $b = [b_1 b_2 .. b_n]$ be two set of data (assume their mean be zero). And we want to find out whether these two are statistically independent? **Covariance** comes into picture

$$\sigma_a^2 = \frac{1}{n-1} a \times a^T \quad \sigma_b^2 = \frac{1}{n-1} b \times b^T \quad \sigma_{ab}^2 = \frac{1}{n-1} a \times b^T$$

Essentially an inner-product

- If a and b are of unit length then, the dot product $a \times b^T$ tells how much they are in same direction.
- If they are in same direction the value would be 1 and when they are orthogonal it would be 0.

Mean, Variance and Covariance

- Consider covariance among all pair of data vectors

$$C_X = \frac{1}{n-1} XX^T = \begin{bmatrix} \sigma_{a_1 a_1}^2 & \sigma_{a_1 a_2}^2 & \cdots & \sigma_{a_1 a_n}^2 \\ \sigma_{a_2 a_1}^2 & \sigma_{a_2 a_2}^2 & \cdots & \sigma_{a_2 a_n}^2 \\ \vdots & \vdots & \cdots & \vdots \\ \sigma_{a_n a_1}^2 & \sigma_{a_n a_2}^2 & \cdots & \sigma_{a_n a_n}^2 \end{bmatrix}$$

- Diagonal has variance and off diagonal covariance. It is a symmetric matrix
- If vectors are in same direction the covariance would be 1 and when they are orthogonal it would be 0.
- We want small covariance and large variance. *i.e.* large values at diagonal and small at rest of the places.
- If we could make it diagonal then there would be no redundancy

Diagonalization

- Let X be the data matrix, consider XX^T this is a symmetric matrix and self adjoint therefore one can always do **eigen value decomposition**.

$$XX^T = S \Lambda S^T$$

where S is matrix of eigen vectors ($S^{-1} = S^T$) and Λ is diagonal matrix of eigen values

- Consider $Y = S^T X$, then

$$C_Y = \frac{1}{n-1} YY^T = \frac{1}{n-1} S^T X (S^T X)^T = \frac{1}{n-1} S^T XX^T S$$

$$C_Y = \frac{1}{n-1} S^T S \Lambda S^T S = \frac{1}{n-1} \Lambda$$

- Wow! covariance matrix of $S^T X$ is diagonal :)

PCA

- Similar operation can also be done by using SVD
- We know every matrix can be written as $X = U \Sigma V^T$
- Let $Y = U^T X$

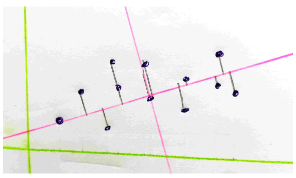
$$C_Y = \frac{1}{n-1} YY^T = \frac{1}{n-1} U^T X (U^T X)^T = \frac{1}{n-1} U^T XX^T U$$

$$C_Y = \frac{1}{n-1} U^T (U \Sigma^2 U^T) U = \frac{1}{n-1} \Sigma^2$$

- Note that $\Sigma^2 = \Lambda$, there is a connection between singular value and eigen value.

Data and dimensionality reduction

Effect of rotation and translation



- We can discover a better representation where there is more spread in the data therefore less chance of misclassification.

See some data [1]

Use *libraOffice* to create a .CSV file. First column is serial number and next three columns get random values from `FLOOR(RAND()*100)`.

1	47	21	22
2	32	80	90
3	35	39	53
4	8	63	73
5	34	10	79
6	1	14	53
7	75	58	39
8	45	81	53
9	71	42	65
10	15	97	70
11	66	84	36
⋮	⋮	⋮	⋮
19	36	27	8
20	7	36	49
21	38	96	93
22	87	71	2
23	52	96	35
24	15	49	34
25	26	98	36
26	81	17	17
27	67	41	70
28	38	91	34
29	11	36	39
30	71	85	91

```
import pandas as pd
from sklearn.decomposition import PCA

df = pd.read_csv(''./myData.csv'', names=['v1','v2','v3','v4'])

for i in range(1,5):
    pca = PCA(n_components=i)
    pca.fit(df)
    print sum(pca.explained_variance_ratio)
```

Result

0.389843872091
0.731978159735
0.96974472518
1.0

Four component take 100% of variation

See some more data [2]

```
import numpy.random as np
numPoints=15
np.seed(500)
v1 = [np.randint(low=1,high=80) for i in range(numPoints)]
v2 = [2*v1[i] for i in range(numPoints)]
v3 = [np.randint(low=1,high=80) for i in range(numPoints)]
v4 = np.permutation(v1)
v5 = [np.randint(low=0,high=2) for i in range(numPoints)]
aData = list(zip(v1,v2,v3,v4,v5))

print aData

  (56, 112, 20, 18, 1), (66, 132, 14, 73, 0), (18, 36, 56, 56, 1),
  (79, 158, 48, 62, 1), (62, 124, 57, 66, 0), (32, 64, 11, 32, 0),
  (73, 146, 47, 79, 0), (72, 144, 14, 78, 1), (78, 156, 3, 18, 1),
  (18, 36, 60, 19, 0), (18, 36, 61, 72, 0), (42, 84, 15, 18, 0),
  (35, 70, 61, 35, 1), (43, 86, 76, 43, 0), (19, 38, 63, 42, 0)

df = pd.DataFrame(data = aData, columns=['v1', 'v2', 'v3', 'v4', 'v5'])
for i in range(1,6):
    pca = PCA(n_components=i)
    pca.fit(df)
    print sum(pca.explained_variance_ratio)
```

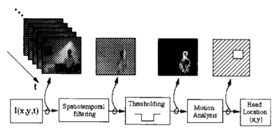
Result

0.759164848209
0.932085855893
0.999943477329
1.0
1.0

Why only four components?
as v_2 is linearly dependent on v_1

PCA in action

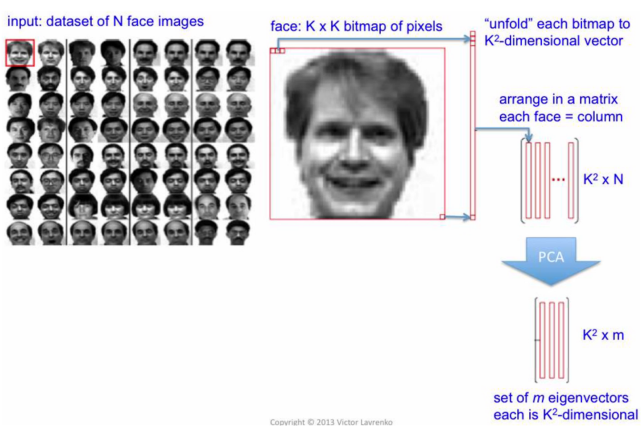
Face recognition (Eigenfaces for recognition [1991] *Turk, Matthew*)



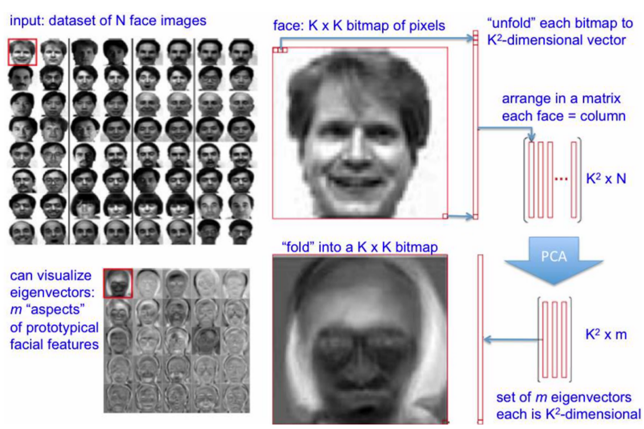
Database of 16 individuals (2500 images) achieved 96% correct classification [1]

[1] https://www.cs.ucsb.edu/ mturk/Papers/jcn.pdf

Eigenfaces



Eigenfaces



Recap: Principal Component Analysis (PCA)

- SVD: $A = U\Sigma V^T$ gives optimal low rank approximation in terms of Frobenius norm $\sqrt{\sum(a_{ij} - b_{ij})^2}$
- Databases are generally of high dimension but true dimensionality may be lot lower than the observed one.
- For **eigen value decomposition** of $XX^T = S\Lambda S^T$. Consider $Y = S^T X$, its covariance matrix $C_Y = \frac{1}{n-1}\Lambda$ is diagonal
- Similar effect can also be obtained using $Y = U^T X$ that is called PCA. It is useful in better representation and dimensionality reduction.
- Eigenfaces achieved 96% accuracy on 2500 images of 16 users

Thank You!

Thank you very much for your attention!

Queries ? Ref[2]

[2] [1] **Face Recognition Using Eigenfaces**, by Turk, CVPR 91